



JÁ VAI COMEÇAR

SQL + Python

ENQUANTO ISSO...

- **Escolha um lugar confortável** para você sentar e se acomodar
- **Que tal pegar um snack** pra matar a fome, **uma água**, um chá
- **Abra o chat**, envie um “olá” e #sentimento de como chega
- Que tal pegar **caderno e caneta para anotações**

T

QUE BOM QUE VOCÊ VEIO!

WELCOME





Fontes de Dados

HENRIQUE SANTANA

Data Product Manager @ **Dell Technologies**

- Já atuei como Cientista de Dados
- Dei aulas de Física em Pré-Vestibular
- Fui dono de bar
- Já trabalhei embarcado na véspera de Natal
- Casei numa festa junina

AGENDA

- **Bloco 1: Motivação**
- **Bloco 2: DQL**
- **Bloco 3: Relacionamentos**
- **Bloco 4: Avançado**
- **Bloco 5: Extra**

AGENDA

- **Bloco 1: Motivação**
- **Bloco 2: DQL**
- **Bloco 3: Relacionamentos**
- **Bloco 4: Avançado**
- **Bloco 5: Extra**



Planilhas Eletrônicas

Software que usa tabelas, que permitem armazenamento e apresentação de dados, e realização de cálculos.

Provocação: Quais os principais pontos fracos das planilhas? E por que continuar usando?

A	B	C	D	E	F
NU_ANO_CENSO	CO_ENTIDADE	NO_ENTIDADE	CO_ORGAO_REGIONAL	TP_SITUACAO_FUNCIONAMENTO	DT_ANO_LETIVO_INICIO
2020	11000023	EEEE ABNAEL MACHADO DE LIMA - CENE	9		1 06/02/2020
2020	11000040	EMEIEF PEQUENOS TALENTOS	9		1 06/02/2020
2020	11000058	CENTRO DE ENSINO CLASSE A	9		1 03/02/2020
2020	11000082	CENTRO EDUCACIONAL PRESBITERIANO 15 DE NOVEMBRO	9		1 03/02/2020
2020	11000104	CENTRO EDUC CORA CORALINA	9		1 03/02/2020
2020	11000171	CENTRO EDUCACIONAL MOJUCA	9		1 04/02/2020
2020	11000180	INTERACAO - CURSOS E COLEGIO	9		2
2020	11000198	COLEGIO SAPIENS - UNIDADE JARDIM DAS MANGUEIRAS	9		1 03/02/2020
2020	11000201	EMEF PROF HERBERT DE ALENCAR	9		1 06/02/2020
2020	11000244	COLEGIO DOM BOSCO	9		1 10/02/2020
2020	11000252	COLEGIO SAPIENS - UNIDADE JARDIM AMERICA	9		1 03/02/2020
2020	11000260	COLEGIO TIRADENTES DA POLICIA MILITAR - CTPM I	9		1 06/02/2020
2020	11000295	EEF SANTA MARCELINA	9		1 03/02/2020
2020	11000309	ESCOLA MUNICIPAL DE EDUCACAO INFANTIL MARISE CASTIEL	9		1 10/02/2020
2020	11000317	EEEFM DR JOSE OTINO DE FREITAS	9		1 11/02/2020
2020	11000325	INSTITUTO EVANGELICO DE EDUCACAO PAUL AENIS	9		1 03/02/2020
2020	11000350	COLEGIO ADVENTISTA DE PORTO VELHO	9		1 20/01/2020
2020	11000368	EMEIEF 13 DE MAIO	4		1 27/04/2020

Reflexão: Não existe bala de prata!

Banco de Dados

Em um banco de dados relacional, normalmente cada linha na tabela é um registro com uma ID exclusiva chamada chave. As colunas da tabela contém atributos dos dados e cada registro geralmente tem um valor para cada atributo



***Reflexão: Nunca é no primeiro
Select!***

Query

Uma query é um pedido de uma informação ou de um dado. Esse pedido também pode ser entendido como uma consulta, uma solicitação ou, ainda, uma requisição.



Reflexão: Organize suas ideias e declare exatamente o que você deseja.

Python

Python é uma linguagem de programação de alto nível, interpretada de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991.

```
31 def __init__(self, settings):
32     self.file = None
33     self.fingerprints = set()
34     self.logdupes = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'requests.json'),
39                         'a')
40         self.file.seek(0)
41         self.fingerprints.update(self.retrieve())
42
43     @classmethod
44     def from_settings(cls, settings):
45         debug = settings.getbool('SUPPRESS_DEBUG')
46         return cls(job_dir(settings), debug)
47
48     def request_seen(self, request):
49         fp = self.request_fingerprint(request)
50         if fp in self.fingerprints:
51             return True
52         self.fingerprints.add(fp)
53         if self.file:
54             self.file.write(fp + os.linesep)
55
56     def request_fingerprint(self, request):
57         return request_fingerprint(request)
```



Reflexão: Simples é melhor do que complexo.

AGENDA

- **Bloco 1: Motivação**
- **Bloco 2: DQL**
- **Bloco 3: Relacionamentos**
- **Bloco 4: Avançado**
- **Bloco 5: Extra**

T Subdivisões do SQL

- **DDL** - Data Definition Language - Linguagem de Definição de Dados. São os comandos que interagem com os objetos do banco.
 - São comandos DDL : CREATE, ALTER e DROP
- **DML** - Data Manipulation Language - Linguagem de Manipulação de Dados. São os comandos que interagem com os dados dentro das tabelas.
 - São comandos DML : INSERT, DELETE e UPDATE
- **DQL** - Data Query Language - Linguagem de Consulta de dados. São os comandos de consulta.
 - São comandos DQL : SELECT (é o comando de consulta)

Aqui cabe um parenteses. Em alguns livros o SELECT fica na DML em outros tem esse grupo próprio.
- **DTL** - Data Transaction Language - Linguagem de Transação de Dados. São os comandos para controle de transação.
 - São comandos DTL : BEGIN TRANSACTION, COMMIT E ROLLBACK
- **DCL** - Data Control Language - Linguagem de Controle de Dados. São os comandos para controlar a parte de segurança do banco de dados.
 - São comandos DCL : GRANT, REVOKE E DENY.



DQL

0 comando SELECT

T SELECT



```
SELECT id, name, price  
FROM products
```


T SELECT



```
SELECT id, name, price  
FROM products  
LIMIT 10
```

T SELECT



```
SELECT id, name, price  
FROM products  
WHERE manufacturer = 'Samsung'
```

T SELECT



```
SELECT id, name, price  
FROM products  
WHERE manufacturer = 'Samsung'  
ORDER BY price DESC  
LIMIT 10
```


T SELECT



```
SELECT [COLUNAS] -- Linhas que serão apresentadas
FROM [TABELA] -- Fonte de dados
WHERE [CONDIÇÃO] -- Filtro para linhas
ORDER BY [COLUNA] [ORDEM] -- A-Z (ASC) ou Z-A (DESC)
LIMIT [No DE LINHAS] -- Quantidade de Linhas
```

Raciocínio: *Selecione para mim as seguintes colunas [a, b, c], mas filtre os registros, ou seja, as linhas, por essas condições [x and y]. E para facilitar a minha visualização, ordene por essa coluna [a] do maior para o menor. Opcional: Como serão muitos registros, me mostre apenas os 25 primeiros.*

T SELECT

[**ATIVIDADE**] Acesse a URL a seguir
e mostre uma tabela com 3 clientes
do México. 

Apenas **nome** e **endereço**.

https://www.w3schools.com/sql/try_sql.asp?filename=try_sql_asc

T SELECT



```
SELECT * FROM Customers  
WHERE Country='Mexico'  
LIMIT 3
```

T

SELECT

	Row Number	COLUMNS			
		CustomerID	CustomerName	ContactName	Address
ROWS	1	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222
	2	3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312
	3	13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993
	4	58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321
	5	80	Tortuga Restaurante	Miguel Angel Paolino	Avda. Azteca 123

AGENDA

- **Bloco 1: Motivação**
- **Bloco 2: DQL**
- **Bloco 3: Relacionamentos**
- **Bloco 4: Avançado**
- **Bloco 5: Extra**

A vertical bar with a gradient from bright green at the top to light blue at the bottom.

RELACIONAMENTOS

Chaves e Joins

T

Unicidade

Provocação: O que é um registro duplicado?



T

Unicidade

**Como trazer uma coluna
com informações únicas?**



```
SELECT DISTINCT name  
FROM products
```

T

Chave Primária

Identificação única de um registro.

```
CREATE TABLE products (  
  id      integer NOT NULL PRIMARY KEY,  
  name    text  
)
```

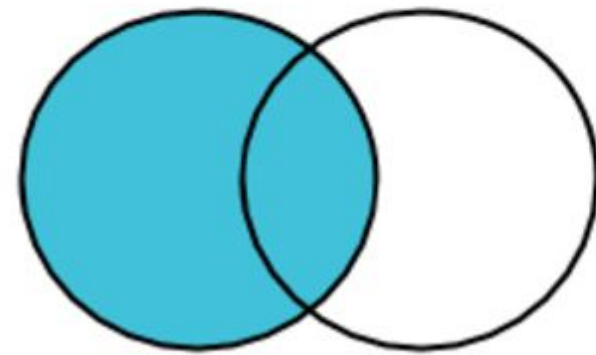
T

Chave Externa (Foreign)

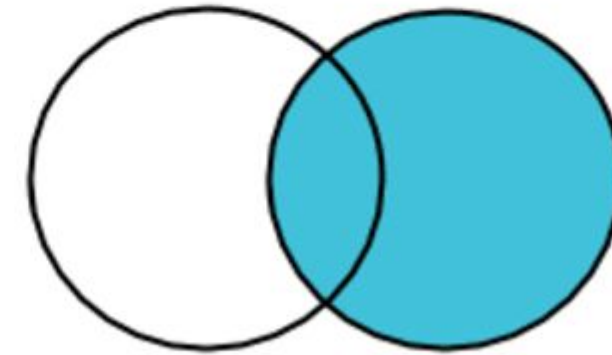
Relacionamento entre tabelas.

```
CREATE TABLE sales (  
  id          integer NOT NULL PRIMARY KEY,  
  sold_at     timestamp,  
  product_id  integer REFERENCES products (id),  
  quantity    integer  
)
```

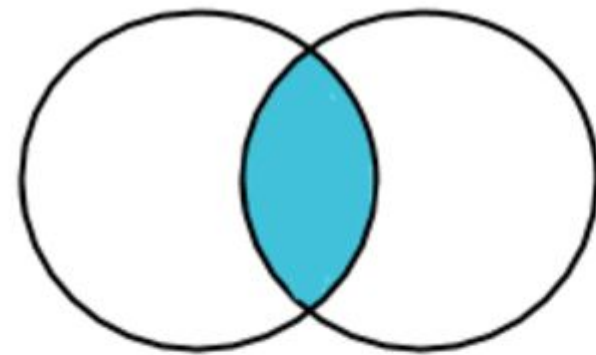
T JOINS



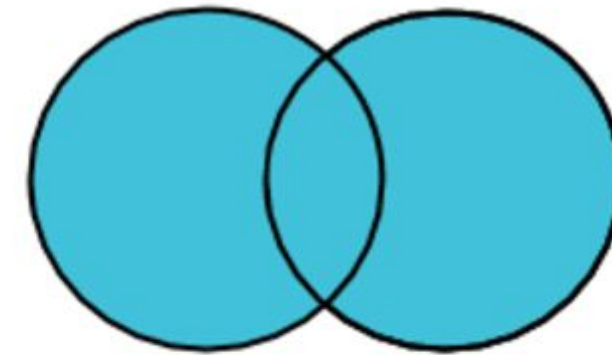
Left Join



Right Join



Inner Join



**Full Outer
Join**

T

Inner Join

**Interseção entre tabelas
baseado em alguma coluna
com dados em comum**



```
SELECT sales.id, products.name  
FROM sales  
INNER JOIN products  
    ON sales.product_id = products.id
```


Left Join

Retorna todos os registros selecionados da tabela da esquerda mais os registros correspondentes da tabela da direita.

```
SELECT products.id, sales.quantity  
FROM products  
LEFT JOIN sales  
    ON products.id = sales.product_id
```

T JOINS

[ATIVIDADE] Acesse a URL a seguir e mostre uma tabela com **Id do Pedido, Data do Pedido e Nome do Cliente**, apenas de clientes de Londres com pedidos feitos a partir de 2017.

https://www.w3schools.com/sql/tryysql.asp?filename=trysql_asc

T JOINS



```
SELECT  Orders.OrderID,  
        Orders.OrderDate,  
        Customers.CustomerName  
FROM Orders  
INNER JOIN Customers  
    ON Orders.CustomerID = Customers.CustomerID  
WHERE Customers.City = 'London'  
    and Orders.OrderDate >= '1997-01-01';
```

Raciocínio: Para cada linha da tabela A, a query a compara com todas as linhas da tabela B. Se um par de linhas fizer com que a condição de junção seja avaliado como *TRUE*, os valores da coluna dessas linhas serão combinados para formar uma nova linha que será incluída no conjunto de resultados.

AGENDA

- **Bloco 1: Motivação**
- **Bloco 2: DQL**
- **Bloco 3: Relacionamentos**
- **Bloco 4: Avançado**
- **Bloco 5: Extra**

ORM

ORM (Object Relational Mapper) é uma técnica de mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam.



SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

T ORM

- Select simples no SQL

```
SELECT * FROM User WHERE name = 'John'
```

- Fazendo queries pythonicas com SQLAlchemy

```
session.query(User).filter(User.name=='John').first()
```

- Usando o SQL Alchemy como Engine para pós-tratamento no Pandas

```
pd.read_sql(query, engine)
```


***Provocação: SQL direto, usar
SQLAlchemy ou Pandas?***



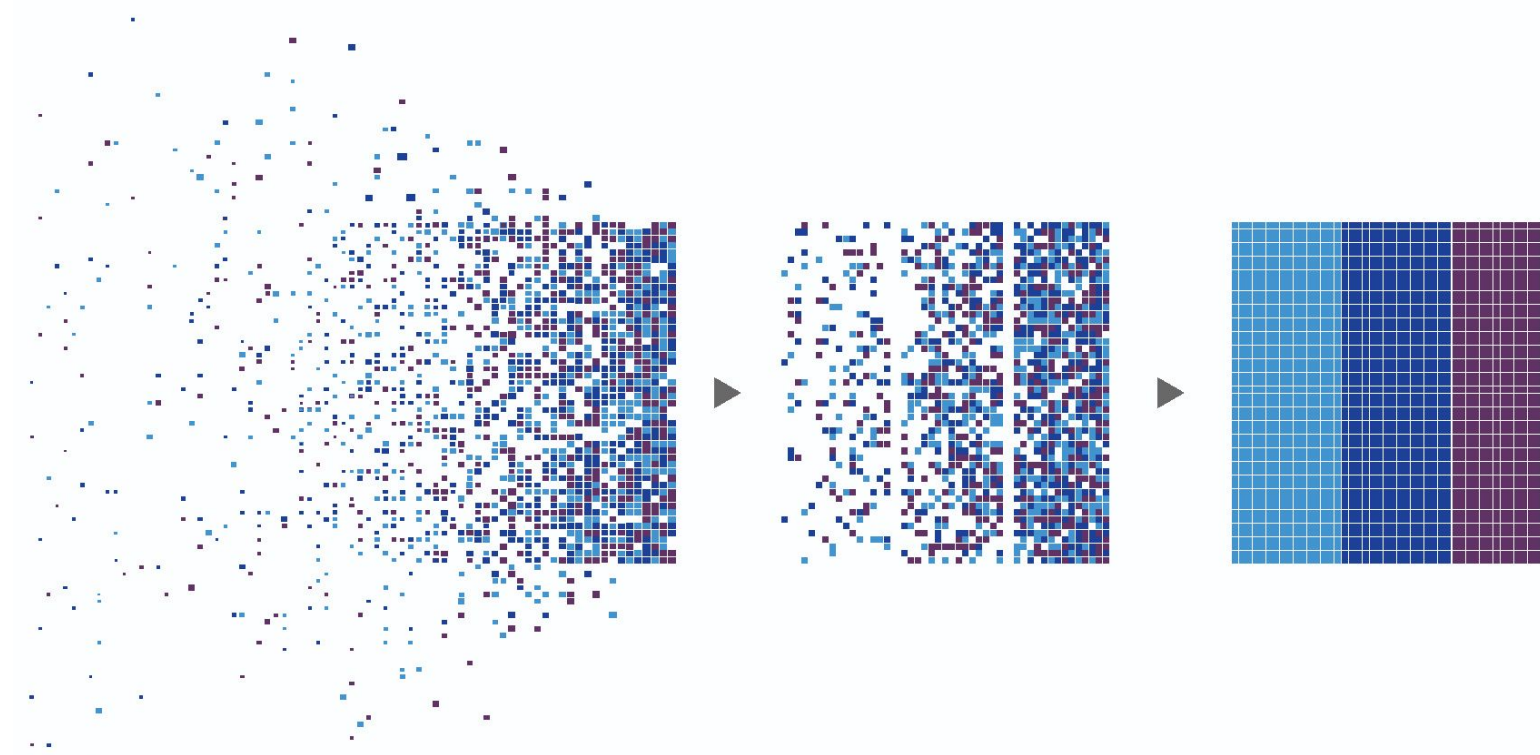
AGREGAÇÃO

Suprimindo informações para ter mais insights

Agregado vs Granular

Granular: Registro de uma venda específica de produto para um cliente.

Agregado: Soma de todas as vendas feitas no último trimestre do ano.



Funções de Agregação



MIN()

SUM()

MIN()

COUNT()

AVG()

Group by

Agrupa as linhas por uma ou mais variáveis (colunas) e solicita uma função de agregação.

```
SELECT products.name,  
       COUNT(sales.id) as count_sales,  
       SUM(sales.quantity) as sum_quantity  
FROM sales  
LEFT JOIN products  
      ON sales.product_id = products.id  
GROUP BY product.name
```

Having

Filtra as linhas após a agregação, isso porque o WHERE não pode ser usado para funções de agregação.

```
SELECT products.name,  
       COUNT(sales.id) as count_sales,  
       SUM(sales.quantity) as sum_quantity  
FROM sales  
LEFT JOIN products  
      ON sales.product_id = products.id  
GROUP BY product.name  
HAVING COUNT(sales.id) > 100
```



ORDEN DE EXECUÇÃO

Como o Banco de Dados executa as suas queries

T ORM

SULIA EVANS
@bork

SQL queries run
in this order

FROM + JOIN



WHERE



GROUP BY



HAVING



SELECT (window functions
happen here!)



ORDER BY



LIMIT



SUBQUERIES, CONTEXTO E VIEWS

Quando a query fica complexa

Subquery

Um comando **SELECT** dentro de outro.

OBS: Funciona dentro de **INSERT**, **UPDATE** e **DELETE** também.

```
SELECT
    film_id,
    title,
    rental_rate
FROM
    film
WHERE
    rental_rate > (
        SELECT
            AVG (rental_rate)
        FROM
            film
    )
```

Subquery

Um comando **SELECT** dentro de outro.

OBS: Funciona dentro de **INSERT**, **UPDATE** e **DELETE** também.

Após o **SELECT** retornando uma coluna

Após o **FROM** como se fosse uma tabela

Dentro de um **WHERE** para fazer comparações

CTE

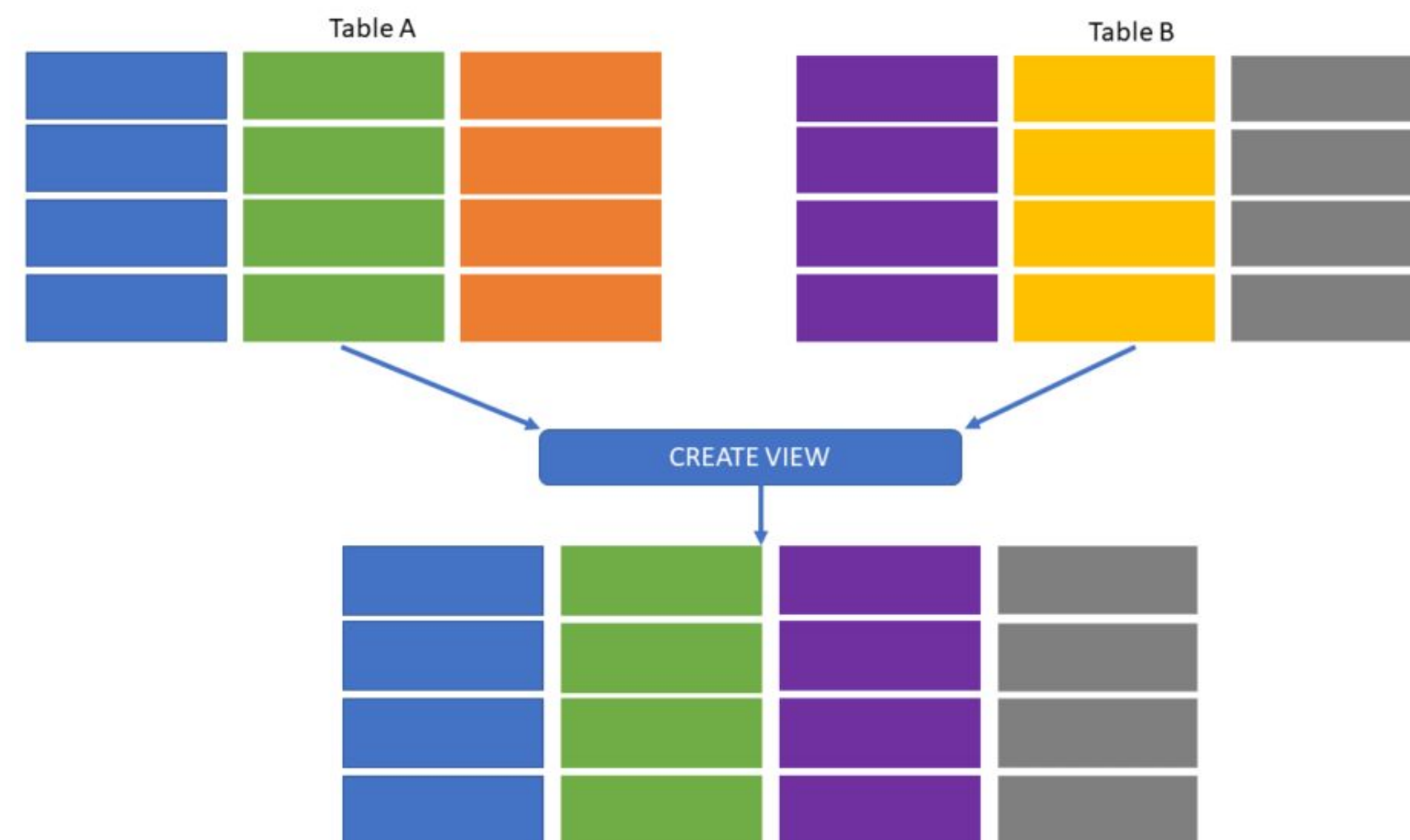
As Common Table Expressions são espécie de tabelas temporárias que ajudam na legibilidade e manutenção de queries complexas.

```
WITH cte_film AS (  
    SELECT  
        film_id,  
        title,  
        (CASE  
            WHEN length < 30 THEN 'Short'  
            WHEN length < 90 THEN 'Medium'  
            ELSE 'Long'  
        END) length  
    FROM  
        film  
)  
SELECT  
    film_id,  
    title,  
    length  
FROM  
    cte_film  
WHERE  
    length = 'Long'  
ORDER BY  
    title;
```

Raciocínio: A minha query está ficando muito complexa. Vou separar em CTEs todas as potenciais tabelas que eu gostaria que já existisse, e dar um nome para cada uma delas. Faço isso para dividir as responsabilidades de processamento, facilitar a compreensão da minha query e possibilitar que eu use essas tabelas em qualquer contexto posterior. O meu último Select vai ser apenas de visualização, ou seja, as colunas que eu realmente quero ver no final.

Views

Uma view é um tipo de tabela virtual que serve para armazenar uma query complexa e apenas retornar os dados.



Views

Uma view é um tipo de tabela virtual que serve para armazenar uma query complexa e apenas retornar os dados.



```
CREATE VIEW V_BR_CUSTOMERS AS  
  SELECT CustomerName, ContactName  
  FROM Customers  
  WHERE Country = 'Brazil'
```

AGENDA

- **Bloco 1: Motivação**
- **Bloco 2: DQL**
- **Bloco 3: Relacionamentos**
- **Bloco 4: Avançado**
- **Bloco 5: Extra**

T



EXTRA

Discussão aberta sobre SQL e Python

T



DÚVIDAS FINAIS

